An Automatic Technique for Checking the Simulation of Timed Systems

Elie Fares, Jean-Paul Bodeveix, M Filali, Manuel Garnacho

An Automatic Technique for Checking the Simulation of Timed Systems

Elie Fares, Jean-Paul Bodeveix, Mamoun Filali-Amine, and Manuel Garnacho

IRIT, Université de Toulouse

Abstract. In this paper, we suggest an automatic technique for checking the timed weak simulation between timed transition systems. The technique is an observation-based method in which two timed transition systems are composed with a timed observer. A μ-calculus property that captures the timed weak simulation is then verified on the result of the composition. An interesting feature of the suggested technique is that it only relies on an untimed μ-calculus model-checker without any specific algorithm needed to analyze the result of the composition. We also show that our simulation relation supports interesting results concerning the trace inclusion and the preservation of linear properties. Finally, the technique is validated using the FIACRE/TINA toolset.

1 Introduction

The verification of real-time systems plays a major role in the design of highly trusted systems. Yet, the more complex the system in terms of space and time is, the less tractable its verification tends to be. Thus, new techniques have been suggested in order to minimize the verification cost in terms of both space and time [16,13,4]. Among these techniques, refinement is one of the most valuable concepts. Roughly speaking, we say that a concrete system C is a proven refinement of an abstract one A, if each time A is used, C can be used instead.

A wide range of refinement relations and sufficient conditions for refinement exist in the literature [31]. Accordingly, one of the most intuitive relations is trace inclusion. However, the problem of timed trace inclusion for non-deterministic systems has been proven to be undecidable if the timed automata model contains at least two clocks [27]. Since abstract specifications often involve non-determinism, this solution is clearly not appropriate. Therefore, the need for a condition that implies the trace inclusion has emerged. Timed simulation relations have been introduced as a sufficient condition for trace inclusion [29]. This class of relations is also decidable [29].

In this paper, we study the problem of automatically proving the timed weak simulation between timed transition systems. First, we start by giving a definition of the timed weak simulation and showing that it implies finite trace inclusion which preserves linear safety properties. We also show that the parallel operator of our constrained timed systems (CTTS see Section 2.2) is also monotonic w.r.t our timed simulation. Second, in order to automatically check

the timed simulation between systems, we follow a standard technique in model checking which is mostly used in the verification of timed properties. The idea is in fact an observation-based method in which A and C are composed with an observer. The result of their composition is then tested using a μ-calculus property that captures the timed weak simulation definition. We show in this paper that for a given class of systems, the μ-calculus criterion is sound and complete. Furthermore, the approach is validated using the FIACRE/TINA toolset [10,7]. We also show that our technique can be used in real life applications by illustrating it on an industrial-inspired example.

To the best of our knowledge, the use of a μ-calculus property in order to verify the simulation in the timed context is new. Furthermore, following our technique, some of the restrictions that exist in the verification of timed weak simulation are relaxed (see Related Work). Another advantage of our approach is that it is self-contained and relies exclusively on existing model checking tools, which means that no specific algorithm for the simulation verification is given.

The paper is organized as follows. In Section 2, we define our behavioral framework which is based on timed transition systems. In Section 3, we briefly recall the syntax and the semantics of the μ-calculus logic. In Section 4, we give our simulation definition along with its properties. We present in Section 5 the core of our verification technique in which we present the observers along with the μ-calculus property. Afterwards, in Section 6, we discuss the experimental results and give an example of the application of the technique before presenting the related work and concluding the paper in Section 7 and Section 8 respectively.

2 Concrete/Abstract Systems

In this section, we present our considered systems. We start by defining the semantic model (Section 2.1) along with the properties it needs to fulfill for the sake of our simulation verification technique. We then give a finite representation of the semantic model (Section 2.2) and some sufficient conditions that imply the properties given at the semantic level.

2.1 Semantic Model

Definition 1 (Timed Transition System TTS). *Let Δ be a time domain [21], e.g., $\mathbb{R}^+$, L a label set containing the silent action τ, a Timed Transition System TTS [9] is a tuple $\langle Q, Q^0, \rightarrow\rangle$ where Q is a set of states, $Q^0 \subseteq Q$ is the set of initial states, and $\rightarrow$ is a transition relation $\subseteq Q \times (L \cup \Delta) \times Q$. We write $q \xrightarrow{l} q'$ for $(q, l, q') \in \rightarrow$. We require standard properties for $\rightarrow$, namely time determinism, reflexivity, additivity, and continuity as defined in [15].*

We define $q \xrightarrow{a^*} q' \triangleq q \xrightarrow{a} q_1 \xrightarrow{a} q_2 \cdots \xrightarrow{a} q'$ and write $q \xrightarrow{ab} q''$ if there exists q' such that $q \xrightarrow{a} q' \xrightarrow{b} q''$ and $q \overset{b}{\Rightarrow} q''$ for $q \xrightarrow{\tau^* b} q'$. We define as well $q \overset{d}{\Longrightarrow}^* q' \triangleq \exists q_0 \xrightarrow{\delta_0} q'_0 \xrightarrow{\tau} q_1 \xrightarrow{\delta_1} q'_1 \xrightarrow{\tau} q_2 \cdots \xrightarrow{\delta_n} q'_n$ such that $\Sigma_{i=0}^{n} \delta_i = d \wedge q = q_0 \wedge q' = q'_n$ and write $q \overset{d}{\Longrightarrow}^+ q'$ when $n > 0$ (there exists at least one τ).

Definition 2 (Timed Trace). *For $\alpha_i \in L - \{\tau\}$ and $\delta_i \in \Delta$, a timed trace is either a finite sequence $((\delta_i \alpha_i)_{i<n})$ or an infinite sequence $((\delta_i \alpha_i)_{i\in\mathbb{N}})$. We denote Tr the set of such traces.*

Definition 3 (TTS Timed Trace). *Given a TTS and I a (finite or infinite) initial segment of $\mathbb{N}$, a timed trace $((\delta_i \alpha_i)_{i\in I})$ is accepted by the TTS if there exists an initial (starting with $q^0 \in Q^0$) TTS execution $((q_i \overset{\delta_i e_i}{\rightarrow} q_{i+1})_{i\in I'})$ where I' is an initial segment of $\mathbb{N}$, $e_i \in L$, and every step in the timed execution corresponds to a transition in the TTS and if I' is finite, the last state has no outgoing transition. The trace is then the sequence of labels of the execution after the elimination of τ events and combination of consecutive δ. We denote by $Traces(T)$ the set of traces of T and $Traces_{fin}(T)$ the set of finite prefixes of elements of $Traces(T)$.*

Definition 4 (τ-Divergence). *Given a set of labels L, a TTS $\langle Q, Q^0, \rightarrow\rangle$ is τ-divergent if for all $q \in Q$ and for all $\delta \in \Delta$, there exists q' such that $q \overset{\delta}{\Longrightarrow}{}^* q'$.*

This means that we require that time can always diverge via τ events. Namely, for all d, there always exists a $\tau\delta$ execution that advances to the date d.

Definition 5 (τ Non-Zeno path). *A TTS is said to have a τ Zeno path if it has an infinite time-convergent execution sequence ($\Sigma_{i=0}^{\infty}\delta_i < \infty$) in which only τ events are executed. A TTS is τ non-Zeno if it does not have such execution sequence, that is all infinite execution sequences of τ actions are time divergent ($\Sigma_{i=0}^{\infty}\delta_i = \infty$).*

The hypothesis of τ non-Zeno will be used to show inductively that a property is preserved through the elapsing of time interleaved with τ transitions.

Lemma 1 (τ Non-Zenoness Characterization). *We give an induction-based definition of the τ non-Zenoness of a TTS [28]. We denote as $P_\delta(s)$ a property P that holds in a state s at time δ. The TTS is τ non-Zeno iff it satisfies :*

$$\frac{\overbrace{P_0(q_0)}^{(1)} \wedge \left(\begin{array}{l} \forall q \in Q\ \forall \delta_2 < \delta_1, q_0 \overset{\delta_2}{\rightarrow} q\ \wedge P_{\delta_2}(q) \Rightarrow \\ \underbrace{\exists q', q \overset{\delta_1-\delta_2}{\rightarrow} q' \wedge P_{\delta_1}(q')}_{(2)} \ \vee\ \overbrace{\exists \delta_3 \in [\delta_2, \delta_1], \exists q', q \overset{\delta_3-\delta_2}{\Longrightarrow}{}^+ q' \wedge P_{\delta_3}(q')}^{(3)} \end{array} \right)}{\exists q', q_0 \overset{\delta_1}{\Longrightarrow}{}^* q' \wedge P_{\delta_1}(q')}$$

The τ non-Zenoness property leads to an induction principle. Here, we say that for a property P to be true in δ, then it is sufficient to show that :

1. P is true at the current instant (1) and,
2. if P is true at a given time, then P must be made true after either a time transition reaching δ (2) or a τ transition (possibly preceded by a delay) (3).

This characterization relies on the fact that time is unbounded ($\forall x \in \Delta, \exists y > x$).

2.2 Constrained Time Transition System (CTTS)

We give the definition of a CTTS which is a syntactic finite representation for the TTS. We also give the properties that need to be satisfied by this finite representation in order to satisfy the assumptions made at the semantic level. A CTTS is close to the abstract model of [20] which introduces the notion of time using time intervals associated to ports.

Definition 6 (Constrained Time Transition System). *Given a set of labels L and the set of intervals $\mathbf{I}$ over the time domain Δ, a CTTS is defined as $\langle Q, Q^0, T, L, \rho : T \to 2^{Q \times Q}, \lambda : T \to L, \iota : T \nrightarrow \mathbf{I}, \triangleright \subseteq T \times T \rangle$ where Q denotes the set of states, $Q^0 \subseteq Q$ is the set of initial states, T denotes the set of transitions, ρ maps each transition to a set of state couples (source,target), λ associates each transition with its label, ι associates a time interval to each transition labeled with a τ (the visible events are not constrained) and $\triangleright$ denotes a time reset relation between two transitions. We write $t : q \xrightarrow{l} q'$ for $(q, q') \in \rho(t) \wedge l = \lambda(t)$.*

We comment on the reset relation $\triangleright$. Each transition of a CTTS is associated to a clock at the semantic level. An enabled transition can be fired when the clock belongs to the time interval of the transition. Whether the firing of a transition resets the clocks of the enabled transitions or not is governed by $\triangleright$: if $t \triangleright t'$, then the firing of t resets the clock of t'. The intuition behind the reset relation stems from the semantics of Time Petri Nets [9]. Based on the intermediary semantics of Time Petri Nets, $t \triangleright t'$ would hold for any pair of transitions sharing an input place. Conversely, based on the atomic semantics, $t \triangleright t'$ only holds for $t = t'$. In our model, we explicitly define the $\triangleright$ relation for each transition.

CTTS Example. In Fig 1, we show how a CTTS is made out of a FIACRE system (High level). The example shows the intuition behind our choice of representing a transition as set of state pairs. We note in the example that t_0 does not reset the clock of t_1. Otherwise, t_1 would never be fired since its lower bound would never be reached.

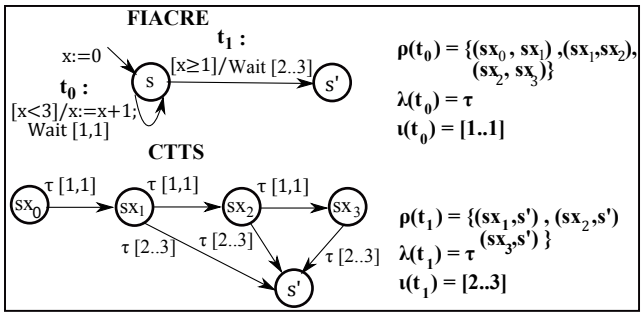

Fig. 1. Example of CTTS

CTTS Semantics. This semantics is defined through three rules: DIS for discrete events, 0-DLY and DLY for time elapse events.

For $(v + \delta)(t) = v(t) + \delta$, $\overleftarrow{I} \triangleq \{x \in \Delta \mid \exists y \in \Delta, x + y \in I\}$ being the downward closure of the interval I and $(q, q') \hookrightarrow t' \triangleq q \notin \mathbf{dom}(\rho(t')) \wedge q' \in \mathbf{dom}(\rho(t'))$ denoting that the transition t' is newly enabled by the transition $q \to q'$, the semantics of a CTTS $\mathcal{T} = \langle Q, Q^0, T, L, \rho, \lambda, \iota, \triangleright \rangle$ is defined as a TTS $[\![\mathcal{T}]\!] = \langle Q \times (T \to \Delta), (Q^0 \times \{t : T \mapsto 0\}), \to \rangle$ such that $\to$ is defined as :

$$\frac{t : q \xrightarrow{l} q',\ v(t) \in \iota(t) \wedge \forall t', v'(t') = \begin{cases} 0 & \textbf{if } (q,q') \hookrightarrow t' \vee t \triangleright t' \\ v(t') & \textbf{else} \end{cases}}{(q,v) \xrightarrow{l} (q',v')}\ \textsc{DiS}$$

$$\frac{}{(q,v) \xrightarrow{0} (q,v)}\ \textsc{0-Dly} \qquad \frac{\forall t \in T, q \in \mathbf{dom}(\rho(t)) \Rightarrow v(t) + \delta \in \overleftarrow{\iota(t)}}{(q,v) \xrightarrow{\delta} (q, v+\delta)}\ \textsc{Dly}$$

Note that the time properties of the TTS are satisfied by the CTTS semantics.

Definition 7 (CTTS Property Satisfaction). *Given a linear temporal formula φ and a CTTS T, we say that T satisfies φ, denoted by $T \models \varphi$, if $\forall tr, (tr \in Traces([\![T]\!]) \Rightarrow tr \models \varphi)$.*

Thus, a CTTS satisfies the property φ if all its traces satisfy φ.

Definition 8 (CTTS Composition). *Given $CTTS_1 = \langle Q_1, Q_1^0, T_1, L, \rho_1, \lambda_1, \iota_1, \triangleright_1 \rangle$, $CTTS_2 = \langle Q_2, Q_2^0, T_2, L, \rho_2, \lambda_2, \iota_2, \triangleright_2 \rangle$ and a set of labels $S \subseteq L$, their composition* [1] *$CTTS_1 \underset{S}{\|} CTTS_2$ is defined as $\langle Q_1 \times Q_2, Q_1^0 \times Q_2^0, T, L, \rho, \lambda, \iota, \triangleright \rangle$ where T* [2] *is defined as :*

$$\frac{t_1 : q_1 \xrightarrow{l_1} q_1',\ l_1 \notin S}{t_1 \uparrow_1 : (q_1,q_2) \xrightarrow{l_1} (q_1',q_2)}\ \textsc{Interleaving}_L \qquad \frac{t_2 : q_2 \xrightarrow{l_2} q_2',\ l_2 \notin S}{t_2 \uparrow_2 : (q_1,q_2) \xrightarrow{l_2} (q_1,q_2')}\ \textsc{Interleaving}_R$$

$$\frac{t_1 : q_1 \xrightarrow{l} q_1', t_2 : q_2 \xrightarrow{l} q_2', l \in S}{t_1 \odot t_2 : (q_1,q_2) \xrightarrow{l} (q_1',q_2')}\ \textsc{Synchronous}$$

The visible events are not time constrained. Thus, only the τ events may be associated to time intervals. ι is only defined on τ transitions. The transitions of the resulting CTTS are associated to the same time intervals they had before the application of the composition operation. Formally, this is defined as :

$$\iota(t \uparrow_1) = \iota(t) \textbf{ if } \lambda(t) = \tau \qquad Time_L \qquad \iota(t \uparrow_2) = \iota(t) \textbf{ if } \lambda(t) = \tau \qquad Time_R$$

For $t_i, t_i' \in T_i$ and $i = \{1,2\}$, the composition of the clock-reset relation $\triangleright$ is defined as :

$$\frac{t_i \ \triangleright\ t_i'}{t_i \uparrow_i \ \triangleright\ t_i' \uparrow_i}\ (1) \qquad \frac{t_i \ \triangleright\ t_i'}{t_1 \odot t_2 \ \triangleright\ t_i' \uparrow_i}\ (2) \qquad \frac{t_i \ \triangleright\ t_i'}{t_i \uparrow_i \ \triangleright\ t_1' \odot t_2'}\ (3)$$

$$\frac{t_1 \ \triangleright\ t_1'}{t_1 \odot t_2 \ \triangleright\ t_1' \odot t_2'} \quad \frac{t_2 \ \triangleright\ t_2'}{t_1 \odot t_2 \ \triangleright\ t_1' \odot t_2'}\ (4)$$

The reset-clock rules mean that if before the composition a transition t resets the clock of another transition t', then after the composition the resulting transition made out of t (either by the $SYNCHRONOUS$ rule or by either one of the $INTERLEAVING$ rules) will reset the clock of the t' transition.

[1] We write $\|$ when S=L and $|||$ when $S = \emptyset$.

[2] T is a disjoint union of $T_1 \times T_2 \uplus T_1 \uplus T_2$ with $\odot \uparrow_1 \uparrow_2$ as constructors.

The intuition of the rule (4) is again based on semantics of Time Petri Nets and is illustrated in Fig 2. Consider that the transitions t_1 and t'_1 synchronize with t_2 and t'_2. In this example, t_1 consumes a resource used by t'_1, thus consuming the resource of the composition of t'_1 and t'_2. Consuming a resource in the context of Time Petri Nets is translated to a reset.

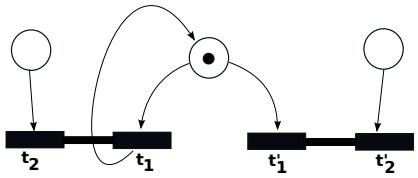

Fig. 2. Reset Composition Rule (4)

Property 1 (Bisimilar States). Given a CTTS T, two states (q, v) and (q, v') in $[\![T]\!]$ that associate the same valuations (w.r.t v and v') to enabled τ transitions are bisimilar.

This means that the states (q, v) and (q, v') can only differ in the valuation associated to τ transitions that are not enabled in q. However the valuations of clocks associated to transitions labeled by visible events can differ because they are unconstrained. The proof of this property is given in [18].

Definition 9 (1-τ). *A CTTS is called 1-τ if it does not have two successive τ actions. Formally,* $\forall t, t'\ t : q \xrightarrow{\tau} q' \wedge t' : q' \xrightarrow{l} q'' \Rightarrow l \neq \tau$.

Definition 10 (Upper Closure). *A CTTS is called upper bounded closed if its upper bounded intervals are closed .*

Property 2 (Upper Closure Preservation). Given two upper bounded closed $CTTS_1$ and $CTTS_2$ and a set of synchronization labels S, their composition $CTTS_1 \underset{S}{\|} CTTS_2$ is also upper bounded closed.

3 μ-Calculus

In this section, we present the μ-Calculus logic. The use of this logic is motivated by its ability to naturally express the definition of various notions of untimed simulations [19]. This cannot be done in other logics containing similar operators and quantifiers like CTL [17].

μ-Calculus Syntax. Let Var be a set of variable names, denoted by Z, Y, ...; let Prop be a set of atomic propositions, typically denoted by P, Q,...; and let L be a set of labels, typically denoted by a, b, ... The set of μ-calculus (L_μ) [11] formulas (w.r.t. Var, Prop,L) is defined as $\varphi ::= \top \mid P \mid Z \mid \varphi_1 \wedge \varphi_2 \mid [a]\varphi \mid \neg\varphi \mid \nu Z.\varphi$. Dual operators are derived, mainly : $\langle a\rangle\varphi \equiv \neg[a]\neg\varphi$ and $\mu Z.\varphi(Z) \equiv \neg\nu Z.\neg\varphi(Z)$. The meaning of $[a]\varphi$ is that φ holds after all a-actions.

μ-Calculus Semantics. The models for the μ-calculus are defined over a structure $\mathfrak{S}$ of the form $\langle S, L, T, v\rangle$ where $\langle S, L, T\rangle$ is a labeled transition system and $v : Prop \rightarrow 2^S$ is a valuation function that maps each atomic proposition $P \in$ Prop to sets of states where P holds. Given a structure $\mathfrak{S}$ and a function $\mathfrak{V} : Var \rightarrow 2^S$ that maps the variables to sets of states in the transition system, the set $\|\varphi\|^{\mathfrak{S}}_{\mathfrak{V}}$ of states satisfying a formula φ is defined as follows :

- $\|\top\|^{\mathfrak{S}}_{\mathfrak{V}} = S$, $\|P\|^{\mathfrak{S}}_{\mathfrak{V}} = v(P)$, $\|X\|^{\mathfrak{S}}_{\mathfrak{V}} = \mathfrak{V}(X)$, $\|\neg\varphi\|^{\mathfrak{S}}_{\mathfrak{V}} = \mathrm{S}$ - $\|\varphi\|^{\mathfrak{S}}_{\mathfrak{V}}$.
- $\|\varphi_1 \wedge \varphi_2\|^{\mathfrak{S}}_{\mathfrak{V}} = \|\varphi_1\|^{\mathfrak{S}}_{\mathfrak{V}} \cap \|\varphi_2\|^{\mathfrak{S}}_{\mathfrak{V}}$.
- $\|[a]\varphi\|^{\mathfrak{S}}_{\mathfrak{V}} = \{s \mid \forall t,\, s \xrightarrow{a} t \Rightarrow t \in \|\varphi\|^{\mathfrak{S}}_{\mathfrak{V}}\}$.
- $\|\nu X.\varphi\|^{\mathfrak{S}}_{\mathfrak{V}} = \bigcup\{Q \in 2^S \mid Q \subseteq \|\varphi\|^{\mathfrak{S}}_{\mathfrak{V}[X \mapsto Q]}\}$ where $\mathfrak{V}[X \mapsto Q]$ is the valuation which maps X to Q and otherwise agrees with $\mathfrak{V}$.

We define the notation $\mathbf{EF}_L P = \mu Z.P \vee \bigvee_{l \in L} \langle l \rangle Z$. This is read as there exists (expressed by $\langle l \rangle$) a finite path labeled by elements of L after which a state is reached where P holds.

4 Timed Weak Simulation and Its Properties

Definition 11 (Timed Weak Simulation). *Given the set of labels L and two TTS $A = \langle Q_a, Q_a^0, \rightarrow_a \rangle$ and $C = \langle Q_c, Q_c^0, \rightarrow_c \rangle$ defined over L, a timed weak simulation between them $\precsim$ is the largest relation such that :*

$$\forall q_c\ q_a, q_c \precsim q_a \Rightarrow$$

E. $\forall q_c'\ e, q_c \xrightarrow{e}_c q_c' \Rightarrow \exists q_a', q_a \stackrel{e}{\Rightarrow}_a q_a' \wedge q_c' \precsim q_a'$ *(Visible Events)*
T. $\forall q_c', q_c \xrightarrow{\tau}_c q_c' \Rightarrow q_c' \precsim q_a$ *(τ Events)*
D. $\forall q_c'\ \delta, q_c \xrightarrow{\delta}_c q_c' \Rightarrow \exists q_a', q_a \stackrel{\delta}{\Longrightarrow}_a^* q_a' \wedge q_c' \precsim q_a'$ *(Delay)*

We say that $C \precsim A$ if $\forall q_C^0 \in Q_C^0, \exists q_A^0 \in Q_A^0$ such that $(q_C^0, q_A^0) \in \precsim$. We say that a simulation holds between two CTTSs if it holds for their semantics.

Theorem 1 (Trace Inclusion). *Given two CTTSs, A and C, if $C \precsim A$ then $Traces_{fin}([\![C]\!]) \subseteq Traces_{fin}([\![A]\!])$.*

The trace inclusion proof is a standard one in the untimed context. Due to the lack of space here, an extension of this proof is given in [18].

Definition 12 (Safety Properties). *A safety property P is defined as a linear time property such that any trace σ where P does not hold contains a bad prefix. Formally, this is defined as follows [8] :*

$$safety(P) \triangleq \forall \sigma \in Tr, \sigma \not\models P \Rightarrow \exists i \text{ such that } \forall \sigma' \in Tr, \sigma_i = \sigma'_i \Rightarrow \sigma' \not\models P$$

where σ_i is the prefix of size i of σ.

Theorem 2 (Property Preservation). *Given two CTTSs A and C, if $C \precsim A$ then any safety linear time property of A is also a property of C.*

Proof. Let P be a safety property such that $A \models P$. We need to prove that $C \models P$. Let $t_c \in Traces([\![C]\!])$. Suppose that $t_c \not\models P$. As P is a safety property, there exists a finite prefix t of t_c such that for all $t' \in Tr$, if t is a prefix of t' then $t' \not\models P$. As t is a prefix of t_c, $t \in Traces_{fin}([\![C]\!])$. As $C \precsim A$, we have $t \in Traces_{fin}([\![A]\!])$. Thus, there exists a $t_a \in Traces([\![A]\!])$ having t as prefix. By choosing $t' = t_a$ we contradict $A \models P$.

The parallel operator is monotonic w.r.t our simulation. Given a component C inside of a composition $C \parallel C_1 \parallel C_2 ... C_n$ the monotony of $\parallel$ is informally described as : if C is replaced by a component C' such that C' simulates C, then $C' \parallel C_1 \parallel C_2 ... C_n$ simulates $C \parallel C_1 \parallel C_2 ... C_n$. This is described as follows :

Theorem 3 (Simulation Compositionality). *Given the CTTSs A_1, A_2, C_1 and C_2 and the set of labels S, we have $C_1 \precsim A_1 \wedge C_2 \precsim A_2 \Rightarrow C_1 \underset{S}{\|} C_2 \precsim A_1 \underset{S}{\|} A_2$.*

This proof is made by showing that each transition of $C_1 \underset{S}{\|} C_2$ has a corresponding transition in $A_1 \underset{S}{\|} A_2$. A complete proof is given in [18].

5 Weak Simulation Verification

5.1 Composing the Abstract/Concrete Systems

Our technique shares its grounds and features with model-checking techniques. Indeed, the first step consists in composing the abstract with the concrete system. Given an abstract CTTS $A = \langle Q_a, Q_a^0, T_a, L, \rho_a, \lambda_a, \iota_a, \triangleright_a \rangle$ and a concrete one $C = \langle Q_c, Q_c^0, T_c, L, \rho_c, \lambda_c, \iota_c, \triangleright_c \rangle$, the two systems are composed after having renamed the events of the two systems by indexing the abstract(resp. concrete) ones by a (resp. c). The composition is thus made asynchronous (Fig. 3) in order to be able to observe all the transitions of the concrete system and verify whether they are simulated by the abstract system. The synchronous composition is not applicable because unmatched concrete transitions may disappear in the product.

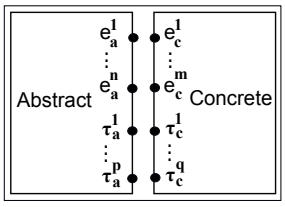

Fig. 3. Systems

5.2 Untimed Weak Simulation Verification

The composition result is analyzed to check the weak simulation between A and C. To do so, the following *Weak Simulation* criterion [19] which corresponds intuitively to the first two rules of the relation $\precsim$ is verified on $A \,|||\, C$:

$$\forall q_a^0 \in Q_a^0 \exists q_c^0 \in Q_c^0, (q_a^0, q_c^0) \models \nu X. \overbrace{\bigwedge_i [e_c^i](\mathbf{EF}_{\tau_a} \langle e_a^i \rangle X)}^{1} \wedge \overbrace{\bigwedge_j [\tau_c^j] X}^{2}$$

(1) means that for each concrete event e_c^i and for each transition labeled by this concrete event e_c^i, there exists a path of a number of abstract local events τ_a that leads eventually to a transition labeled by the abstract event e_a^i such that the target verifies recursively (1) and (2).

(2) means that after each transition labeled with a concrete local event τ_c^j the simulation is maintained.

5.3 Extension to the Timed Context

The already seen property could not be used directly in the timed context since it assumes that concrete and abstract events are composed asynchronously, while the composition of time transitions is necessarily synchronous because time advances at the same rate at the two sides of the composition. Two alternatives are possible. The first is to specify these timing constraints in a timed variant of μ-calculus. The second is to specify the timing aspects with timed observers, to compose the analyzed system with these observers, and to make use of an untimed logic. We follow the second technique. For this purpose, we define two observers (Fig. 4 and 5) :

1. The Control Observer consists in observing the control aspects of the two systems. Namely, for each concrete event, the control observer tries to find a matching abstract event that happens at the same time.
2. The Time Observer models the elapsing of time in the two systems.

In the two observers, the reset relations are empty. This way, the observers never impact the reset relations defined in the abstract and the concrete systems.

Control Observer. The Control Observer is depicted in Fig 4. At the initial state `ok`, the observer synchronizes with either one of the events e_a^i, τ_c^i or e_c^i. When synchronizing with any of e_i^a the observer signals an error (`err` state) since a concrete event is not yet found. When synchronizing with any of τ_c^i the observer maintains the state `ok`. Finally, when synchronizing with the concrete events e_c^i, it tries to match them with the abstract events e_a^i. After a concrete event e_c^i is received, the observer transits to the state $\texttt{wait}_i$ meaning that it now awaits for a matching abstract event e_a^i. At this point, the following scenarios may happen :

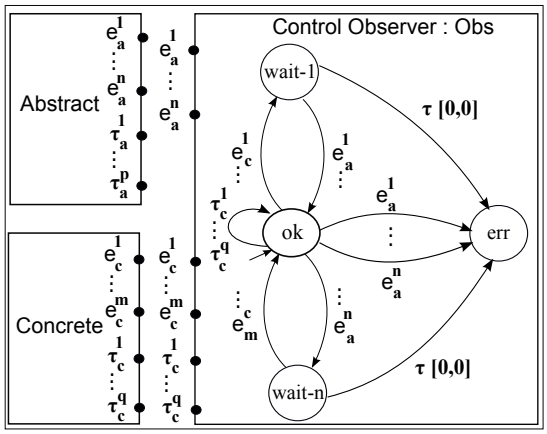

Fig. 4. Control Observer

– A matching abstract event is found and the observer transits back to `ok`.
– The abstract system violates the timing of the concrete system and the observer transits to the state `err`. That is, a matching abstract event is not possible at the same time as the corresponding concrete event. This is modeled by signaling an error in 0 units of time (u.t.). Hence, in case a matching event is found in 0 u.t., we would reach a non-deterministic choice between transiting back to `ok` or also to `err`. The two transitions would then be present in the composition process. This choice is later resolved in the μ-calculus property by searching for a path that satisfies the simulation and thus ignoring the error transition.

Time Observer. The control observer only checks whether two corresponding events could happen simultaneously. However, it mentions nothing about when an elapsing of time occurs. This leads to the definition of an additional observer `Time Observer` (Fig. 5) in which two aspects are modeled. First, at the initial state `evt0`, only the transitions that are firable in 0 time can occur. This is done by specifying a concurrent choice between a timed event τ_0 constrained with $[0, 0]$ and all the events of the abstract and concrete systems. Second, it makes visible the implicit elapsing of time. At the state `evtDly`, on each elapsing of time, a timed event `delay` associated with the constraint $]0, \infty[$ is signaled. This event is later used by the μ-calculus property as an time elapsing marker.

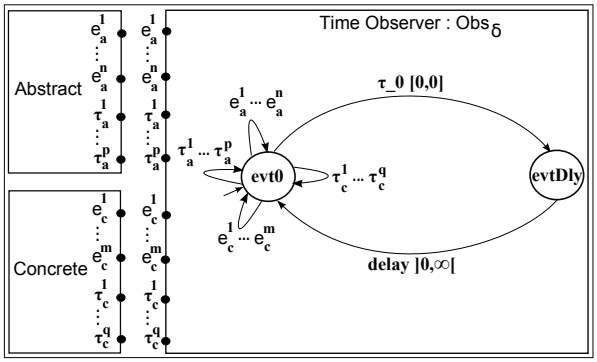

Fig. 5. Time Observer

Assumption 1 (Concrete/Abstract). *A concrete system is any upper bounded closed CTTS (Definition 10). An abstract system is any τ non-Zeno (Definition 5), τ divergent (Definition 4), 1-τ (Definition 9), upper bounded closed CTTS.*

The hypothesis 1-τ is a sufficient condition for the $\tau - \delta$ permutation property.

Definition 13 ($\tau - \delta$ Permutation). *Given a TTS, for all $q\ q'\ q_1\ q_2 \in Q, \delta \in \Delta$, $q \xrightarrow{\delta} q' \wedge q \xrightarrow{\tau} q_1 \xrightarrow{\delta} q_2 \Rightarrow \exists q'', q' \xrightarrow{\tau} q'' \wedge q_2 \sim q''$ where $\sim$ denotes the timed strong bisimulation.*

This means that from a state q, transitions τ and δ may be exchanged leading to bisimilar states q_2 and q''. This property is close to the persistency of [26] that says that time cannot suppress the ability to do an action. However, our requirement that $q_2 \sim q''$ is stronger. This property is not true in general since the clocks newly reset at the state q'' have different values at q_2. The 1-τ hypothesis is a sufficient condition on the CTTSs so that the clock differences at q_2 and q'' would not affect the overall execution of the system.

Theorem 4 (1-τ is a sufficient condition for $\tau - \delta$ permutation). *Given a 1-τ CTTS $\mathcal{T}$, its semantics $[\![\mathcal{T}]\!]$ verifies the $\tau - \delta$ permutation.*

The proof of this theorem is given in [18].

Timed Weak Simulation Verification. The check of timed weak simulation consists in a property verified on the composition of the abstract, the concrete and the observers $(A \ ||| \ C) \ \| \ (Obs \ \| \ Obs_\delta)$ where Obs is the control observer and Obs_δ is the time observer. The simulation of time transitions consists in verifying whether each delay made by the concrete system can also be made

by the abstract one. But unlike the asynchronous composition in the untimed context with which we were able to alternate between the occurrence of the concrete and the abstract events, time is always synchronous in each of A, C and the two observers. Alternating concrete and abstract events does not apply to time transitions. The $TimedWeakSimulation(e_c, e_a, \tau_c, \tau_a)$ criterion is :

$$\overbrace{\forall q_a^0 \in Q_a^0 \exists q_c^0 \in Q_c^0, (q_a^0, q_c^0, ok, evt0) \models \nu X. \overbrace{Obs\ in\ ok \wedge Obs_\delta\ in\ evt0}^{(1)} \wedge}^{(2) Weak\ Simulation}$$
$$\overbrace{\bigwedge_i [e_c^i](\mathbf{EF}_{\tau_a}\langle e_a^i\rangle X) \wedge \bigwedge_j [\tau_c^j] X}^{(2) Weak\ Simulation} \wedge \overbrace{(\mathbf{EF}_{\tau_a}\langle delay\rangle \top) \Rightarrow \mathbf{EF}_{\tau_a}(\langle delay\rangle \top \wedge [delay] X)}^{(3)}$$

This property characterizes a set of product states to which the initial state must belong. This set of states is defined over the composition of states of A,C and the two observers. We comment on the *Timed Weak Simulation* criterion :

- (1) denotes the acceptance of a concrete event at current time.
- (2) is the untimed weak simulation criterion.
- (3) denotes that if time can elapse (delay event) in the product via a sequence of τ abstract events -meaning that time can also elapse in the concrete system since the abstract is τ divergent- then time may elapse and for all possible delay events the simulation holds after a number of τ abstract events. In this part of the formula, $\langle delay\rangle \top$ means that in the current state, it is possible to do a transition labeled with the *delay* event.

The proof of the correctness of the μ-calculus criterion w.r.t the mathematical definition of the timed weak simulation is based on the comparison between two relations defined as the largest relations which can also be seen as the greatest fixed points of monotone set functions. The timed weak simulation criterion is correct without the hypothesis $1 - \tau$. This is why we observe τ sequences. However it is not complete. Due to the lack of space, this proof is given in [18]. It has been also formalized and validated in the proof assistant Coq [30]. The complete Coq proof is found at [3].

Discussion on the Assumptions. We discuss our major restrictions :

1. τ non-Zenoness and τ divergence: these two are standard assumptions made on timed systems. In our context, they guarantee the progress of time in the abstract system. This is necessary in our composition-based method because time is always synchronous. Blocking time in the abstract system could result in blocking time in the whole composition and hide concrete delays.
2. No successive τ transitions : permitting τ transitions in A complicates the verification of the timed weak simulation, because in this case, any delay in C can be matched by a series of delays in A separated by τ transitions [12]. Moreover, with our 1-τ restriction, general modeling techniques of real time systems are still permitted. For instance, specifying an upper bound of a global event e is made by a choice between a timed local event τ and the event e. Specifying a lower bound of a global event e is made by a sequential execution of a timed local event τ and e.

6 Experimental Results

The technique is validated using the FIACRE/TINA toolset (Fig 6). The systems and the observers are written in FIACRE (captured here by a CTTS) and the μ-calculus property is verified on the FIACRE model using the model checker TINA. The FIACRE systems are adapted to the input language of TINA thanks to a translation to TTS. TINA generates a time abstracted automaton, that preserves branching properties, on which the μ-calculus property is verified.

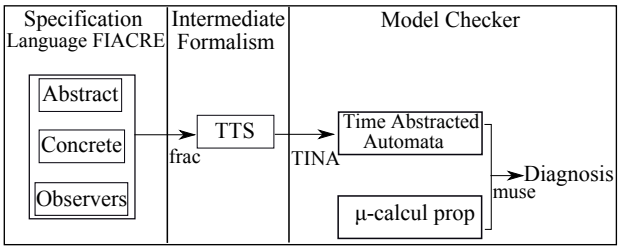

Fig. 6. Approach

6.1 Example of Application

The example is a simplification of an industrial use case that is used to illustrate the BPEL web-services composition language [6].

Abstract and Concrete Modeling. The abstract specification Fig. 7 describes that upon the reception of the purchase request, the process initiates the treatment of the command which takes at most 8 u.t.. Afterwards, either a successful response is sent or the command is simply dropped. A potential refinement of this specification is a composition of three entities (Fig. 8).

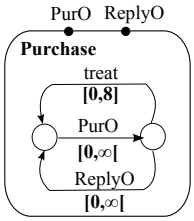

Fig. 7. Specification

Upon receiving a purchase order, the management consults the inventory where the stock is checked in 0 to 4 u.t.. In case of availability of the products, the inventory department sends the shipping price to the financial department before sending the result to management. Management then sends the products price to the financial department where the final price is computed in 2 u.t.. This price is then given to the management and an immediate response is sent to the customer. The transitions of the abstract/concrete system reset the clocks of all the others.

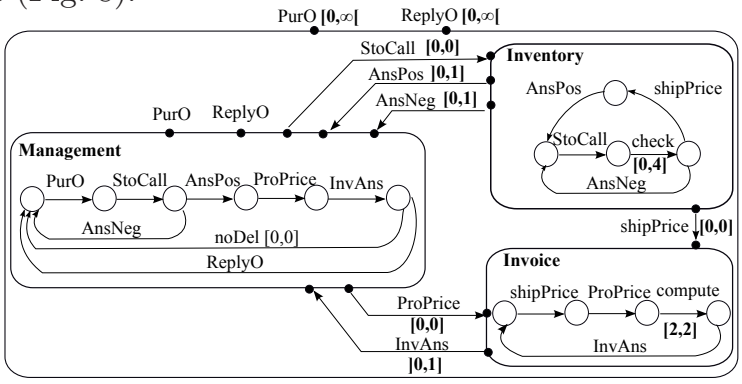

Fig. 8. Purchase Treatment

Simulation Verification. After renaming the events of the systems ($PurO_a$, $ReplyO_a$, $PurO_c$, $ReplyO_c$, ...), they are composed with the two observers. The composition process results in 184 states and 694 transitions. This process is then model checked by an instantiation of the timed weak simulation criterion 5.3 with $e_a = \{PurO_a, ReplyO_a\}$, $e_c = \{PurO_c, ReplyO_c\}$, $\tau_a = \{treat_a\}$ and $\tau_c = \{ShipPrice, ProPrice, InvAns, compute, StoCall, AnsPos, AnsNeg, check, noDel\}$.

The set of states returned by the property contains the initial state of the process. The simulation is then verified. Now suppose that the time interval of `compute` is changed to $[3, 3]$. In this case, the verification of the μ-calculus property does not hold. This is because the concrete system violates the time allowed by the specification. Finally, the example may be found at [2].

7 Related Work

Even though the study of simulation relations have reached a mature level, timed simulation verification is still an open research subject. Interesting results have been elaborated for different timed formalisms ranging from timed transition systems, to timed automata [5,25], and timed input output automata TIOA [23,14]. However, a less tackled aspect of this research is the automatic verification of timed simulations and especially timed weak simulations. A work which resembles ours appears in [22] in the context of the Uppaal [24] tool. A timed simulation for fully observable and deterministic abstract models is reduced to a reachability problem. This is done via a composition with a testing automaton monitoring whether the behavior of the concrete system is within the bounds of the abstract system and then by checking whether an undesired state is never reached. Compared to this result, we do not restrict our abstract systems to deterministic ones. Furthermore, our abstract systems are not fully observable.

Probably the most complete work is the one of [12,14] which led to the ECDAR tool [1]. In this tool, a timed (weak) simulation verification between two TIOAs is supported. The verification is done via a game-based algorithm between the abstract and the concrete systems [12]. Clearly, a TIOA is different in nature from timed transitions systems regarding its input/output semantics. However, their restriction that the abstract systems does not have any τ actions is relaxed in our technique to no successive τ actions. Moreover, our restriction to upper bounded closed CTTSs can be found in their formalism in the form of restricting the TIOAs states invariants constraints to $clock \leq constant$. Finally, unlike theirs, in our technique no specific algorithm is written to analyze the result of the abstract/concrete composition.

Finally, μ-calculus properties were used as a mechanism to capture and to check simulation relations in the untimed context [19]. The Mec 5 model checker, which handles specifications written in Altarica, embeds the μ-calculus logic as a support for properties verifications on Altarica models. This allows users to check weak/strong simulation and bisimulation relations.

8 Conclusion

We have presented an automatic technique for checking the timed weak simulation between timed transition systems originating from CTTS systems. The technique is based on the idea of composing the analyzed systems with observers and then model check their result using a μ-calculus property which captures the timed weak simulation definition. To the best of our knowledge, this is an original approach. Our criterion is sound and complete for a subclass of timed systems. This, along with all the paper results, have been proven using Coq.

Due to the lack of space and for clarification purposes, we applied our technique on a rather simple, but yet a complete example. For the interested readers, an illustration of the technique is made on a more elaborated example in [18] where the technique is adopted to prove the simulation between FIACRE systems translated from BPEL. The initial process consists of around 7K states and 15K transitions with an execution time of 7 seconds while the product process results in around 290K states and 874K transitions with an execution time of 70 seconds. The verification time of the μ-calculus property is 27 seconds.

On another matter, the specification of the observers and the property is manual for now. However, an automatic generation of these two is obtained directly from the alphabet of the processes..

For our future work, we are currently looking into eliminating the 1-τ restriction. Another complementary work pertains to extending our simulation so that it preserves all linear time properties. Finally, it would be insightful to investigate whether the theoretical results for timed automata can be applied to FIACRE systems (CTTS), or whether our simulation verification technique can be applied to timed automata. Such a study is promising since both CTTS and Timed Automata rely on the same semantical framework.

References

1. `http://ecdar.cs.aau.dk/`
2. `http://www.irit.fr/~Elie.Fares/fiacre/refinement/`
3. `http://www.irit.fr/~Jean-Paul.Bodeveix/COQ/Refinement/`
4. Alur, R., Dang, T., Ivančić, F.: Predicate abstraction for reachability analysis of hybrid systems. ACM Trans. Embed. Comput. Syst. 5(1), 152–199 (2006)
5. Alur, R., Dill, D.L.: A theory of timed automata. Theor. Comput. Sci. 126(2), 183–235 (1994)
6. Alves, A., Arkin, A., Askary, S., Bloch, B., Curbera, F., Goland, Y., Kartha, N., Sterling, König, D., Mehta, V., Thatte, S., van der Rijn, D., Yendluri, P., Yiu, A.: Web Services Business Process Execution Language Version 2.0. OASIS (May 2006)
7. Berthomieu, F.V.B., Ribet, P.-O.: The tool tina – construction of abstract state spaces for petri nets and time petri nets. International Journal of Production Research 42 (2004)
8. Baier, C., Katoen, J.-P.: Principles of Model Checking (Representation and Mind Series). The MIT Press (2008)

9. Bérard, B., Cassez, F., Haddad, S., Lime, D., Roux, O.H.: Comparison of different semantics for time Petri nets. In: Peled, D.A., Tsay, Y.-K. (eds.) ATVA 2005. LNCS, vol. 3707, pp. 293–307. Springer, Heidelberg (2005)
10. Berthomieu, B., Bodeveix, J.-P., Farail, P., Filali, M., Garavel, H., Gaufillet, P., Lang, F., Vernadat, F.: Fiacre: An Intermediate Language for Model Verification in the Topcased Environment. In: ERTS 2008, Toulouse, France (2008)
11. Bradfield, J., Stirling, C.: Modal mu-calculi. In: Handbook of Modal Logic, pp. 721–756. Elsevier (2007)
12. Bulychev, P., Chatain, T., David, A., Larsen, K.G.: Efficient on-the-fly algorithm for checking alternating timed simulation. In: Ouaknine, J., Vaandrager, F.W. (eds.) FORMATS 2009. LNCS, vol. 5813, pp. 73–87. Springer, Heidelberg (2009)
13. Clarke, E.M., Grumberg, O., Long, D.E.: Model checking and abstraction. ACM Trans. Program. Lang. Syst. 16(5), 1512–1542 (1994)
14. David, A., Larsen, K.G., Legay, A., Nyman, U., Wąsowski, A.: Methodologies for specification of real-time systems using timed I/O automata. In: de Boer, F.S., Bonsangue, M.M., Hallerstede, S., Leuschel, M. (eds.) FMCO 2009. LNCS, vol. 6286, pp. 290–310. Springer, Heidelberg (2010)
15. David, A., Larsen, K.G., Legay, A., Nyman, U., Wasowski, A.: Timed I/O automata: a complete specification theory for real-time systems. In: Proc. of HSCC 2010, pp. 91–100. ACM, New York (2010)
16. Dierks, H., Kupferschmid, S., Larsen, K.G.: Automatic abstraction refinement for timed automata. In: Raskin, J.-F., Thiagarajan, P.S. (eds.) FORMATS 2007. LNCS, vol. 4763, pp. 114–129. Springer, Heidelberg (2007)
17. Emerson, E.A.: Model checking and the mu-calculus. DIMACS Series in Discrete Mathematics American Mathematical Society, pp. 185–214. American Mathematical Society (1997)
18. Fares, E., Bodeveix, J.-P., Filali, M.: An automatic technique for checking the simulation of timed systems. Technical Report IRIT/RT–2013-10–FR (January 2013), `http://www.irit.fr/~Elie.Fares/PUBLICATIONS/RTIRIT--2013-10--FR.pdf`
19. Griffault, A., Vincent, A.: The mec 5 model-checker. In: Alur, R., Peled, D.A. (eds.) CAV 2004. LNCS, vol. 3114, pp. 488–491. Springer, Heidelberg (2004)
20. Henzinger, T., Manna, Z., Pnueli, A.: Timed transition systems. In: de Bakker, J., Huizing, C., de Roever, W., Rozenberg, G. (eds.) REX 1991. LNCS, vol. 600, pp. 226–251. Springer, Heidelberg (1992)
21. Jeffrey, A.S., Schneider, S.A., Vaandrager, F.W.: A comparison of additivity axioms in timed transition systems. Technical report, Amsterdam, The Netherlands, The Netherlands (1993)
22. Jensen, H.E., Guldstr, K., Skou, A.: Scaling up uppaal: Automatic verification of real-time systems using compositionality and abstraction. In: Joseph, M. (ed.) FTRTFT 2000. LNCS, vol. 1926, pp. 19–30. Springer, Heidelberg (2000)
23. Kaynar, D.K., Lynch, N., Segala, R., Vaandrager, F.: The Theory of Timed I/O Automata (Synthesis Lectures in Computer Science) (2006)
24. Larsen, K.G., Pettersson, P., Yi, W.: Uppaal in a nutshell. Int. Journal on Software Tools for Technology Transfer 1, 134–152 (1997)
25. Lynch, N., Vandraager, F.: Forward and backward simulations - part ii: Timing-based systems. Information and Computation 128 (1995)
26. Nicollin, X., Sifakis, J.: An overview and synthesis on timed process algebras. In: Huizing, C., de Bakker, J.W., Rozenberg, G., de Roever, W.-P. (eds.) REX 1991. LNCS, vol. 600, pp. 526–548. Springer, Heidelberg (1992)
27. Ouaknine, J., Worrell, J.: On the language inclusion problem for timed automata: Closing a decidability gap. In: LICS, pp. 54–63. IEEE Computer Society (2004)

28. Schneider, S.: Concurrent and Real Time Systems: The CSP Approach, 1st edn. John Wiley & Sons, Inc., New York (1999)
29. Tasiran, S., Alur, R., Kurshan, R.P., Brayton, R.K.: Verifying abstractions of timed systems. In: Sassone, V., Montanari, U. (eds.) CONCUR 1996. LNCS, vol. 1119, pp. 546–562. Springer, Heidelberg (1996)
30. C. D. Team. The Coq proof assistant reference manual, version 8.2 (August. 2009)
31. van Glabbeek, R.J.: The linear time-branching time spectrum (extended abstract). In: Baeten, J.C.M., Klop, J.W. (eds.) CONCUR 1990. LNCS, vol. 458, pp. 278–297. Springer, Heidelberg (1990)